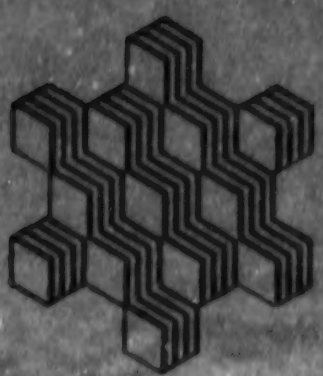
16-0081A

SECONDARY SCHOOL CURRICULUM

INTRODUCTION TO COMPUTER SCIENCE



Québec

Niveau(x) d'enseignement: 4 Numéro EDUQ : 6870797
Centre d'intérêt: 046

Introduction to computer science / Direction générale du développement pédagogique, Direction de la formation générale. -- [Québec] : Ministère de l'éducation, c1984.
66 p. ; 28 cm. -- (Secondary school curriculum)

Code: 16-00618 ; 8384-0936. Bibliogr.: p. 57-59. Glossaire.

ISBN 2-550-06299-X

Vedette(s)-secondaire(s): Québec (Province) Direction générale du développement pédagogique. Direction de la formation générale

Descripteur(s): Initiation à l'informatique * Programme d'enseignement * Secondaire (Ordre d'enseignement)

Le document présente le programme d'enseignement de l'initiation à l'informatique pour le secondaire 4 et 5. L'introduction justifie le programme, indique les relations entre ce programme particulier et les objectifs généreux de l'enseignement au secondaire, montre la continuité du programme avec le programme intitulé "Introduction au langage de programmation", indique la population cible, précise l'organisation des cours et les conditions spécifiques d'implantation du programme. La deuxième partie indique les orientations générales d'implantation du programme. La troisième partie décrit les objectifs généraux et intermédiaires du programme, tandis que la quatrième partie traite de la finalité et des méthodes d'évaluation du programme. La cinquième partie aborde deux questions particulières: les conditions d'admission et le temps à allouer à ce programme. Une bibliographie sommaire est fournie en appendice, ainsi qu'un glossaire d'informatique.

Type de document: d

EDUQ
8870787

INTRODUCTION TO COMPUTER SCIENCE

Direction générale du développement pédagogique
Direction de la formation générale

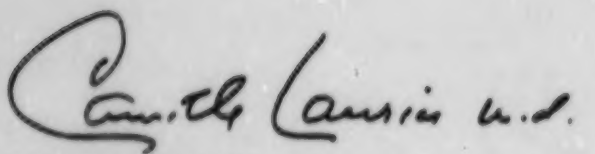
Approuvé par les comités protestant et catholique
du Conseil supérieur de l'éducation
les 20 et 25, 28 novembre 1981

© Gouvernement du Québec
Ministère de l'Éducation, 1984

ISBN 2-550-06299-X

Dépôt légal - premier trimestre 1984

I am pleased to confirm that the program entitled *Introduction to Computer Science*, issued in conformity with Section 3 of the *Regulation Respecting the Basis of Secondary School Organization*, has received the approval of the Confessional Committees of the Conseil supérieur de l'éducation and constitutes an official program which I authorize for use in all schools beginning July 1, 1983.



Camille Laurin, M.D.
Minister of Education

TABLE OF CONTENTS

1. **Introduction**

1.1 Justification for the Program 3

1.2 Relationship Between the Program and the General Objectives of Secondary Education: 4

(A) To develop the pupil's autonomy by introducing him to the thought processes which characterize computer science; 4

(B) To help the pupil to understand technological changes and the many applications of the computer in a constantly evolving world 5

1.3 Continuity Between the Program and "Introduction to a Programming Language" 7

1.4 Target Populations 7

1.5 Course Organization 8

1.6 Specific Conditions of Implementation 8

2. **General Orientations of the Program**

2.1 Socio-Cultural Dimension 11

2.2 Psychoeducational Dimension 12

(A) Major Concerns 12

(B) Instructional Approach 14

3. **Objectives**

3.1 General Objectives 21

3.2 General and Terminal Objectives 21

3.3 General, Terminal, and Intermediate Objectives 24

4. **Evaluation**

4.1 Functions of Evaluation 47

4.2 Subjects of Evaluation 48

4.3 Recommendations 49

5. Special Factors

5.1 Conditions of Admission 55

5.2 Time Allocation 56

6. Bibliography 57

7. Glossary 61

1
Introduction

1. Introduction

1.1 Justification for the Program

The use of the computer is becoming increasingly widespread in contemporary society, in Québec as well as in the rest of the world. Computers directly or indirectly affect many aspects of everyday life, particularly the occupational activities of a growing portion of the population.

Computers are used for billing by public utilities, for keeping up-to-date financial and banking records, for computing students' grades, for planning pupils' schedules, for making certain predictions such as election results and stock market trends. Computers are found more and more frequently in the manufacturing and industrial sectors as well as in the public and parapublic sectors. In the scientific and technological community they are invaluable tools in research and development activities. The field of education also has been affected by the computer. Computer science is taught in the universities, colleges, and school boards. Computer-Assisted Instruction plays an increasingly important role in the educational system. It should be pointed out, as well, that over the years several school boards have acquired a great deal of valuable experience with respect to the applications of the computer in education. Obviously, certain needs exist which call for action on the part of the ministère de l'Éducation and its partners. Finally, in a different but related context, computers are put to all kinds of personal uses by the general public: games, family bookkeeping, control of home appliances and security devices, language learning, etc.

This list of uses or applications of the computer could be more extensive. However, these few examples are enough to illustrate the "invasion" of our culture by computers and information processing. Yet, while Québec is not unique in this respect, it appears that the great majority of Québecers have no definite idea about the place of the computer in their environment, and still less about its functions and uses. The individual who is ignorant about computers finds himself in a position of vulnerability and inequality. In order to rectify this situation, it is necessary to take steps to introduce the population in general and the school population in particular to information processing so that every citizen will know the value and capabilities of the computer. This forward-looking approach should make it possible for an increasing number of Quebec women and men to better understand their present and future environment.

It is obvious that in his education the pupil should find whatever is useful and necessary to his progress. His needs in terms of knowledge can be satisfied in part by computer science. The methodology, basic principles, and processes of this discipline should provide him with certain intellectual tools and techniques which will be most helpful to him in other fields of learning.

It is for the sake of the future that concrete steps should be taken to remedy the general lack of knowledge and understanding of the computer and information processing. It is important to realize that in this endeavour it is not possible to reach every individual concerned in a single attempt. The underlying aim of this Computer Science Program is to "usher in a new era." Consequently, the development and implementation of the program should be handled carefully so that it can be adjusted and modified to suit any situation. The program should be implemented in a gradual and controlled manner, after having sensitized people to its content and established the evaluation format.

1.2 Relationship Between the Program and the General Objectives of Secondary Education

Aims and Values: In *The Schools of Québec, Policy Statement and Plan of Action*, the ministère de l'Éducation defined several aims for the schools of Québec.[1] Among them:

— developing the whole person,

— making education suitable to all.

Overall Objectives: These aims and values are translated into a number of educational goals and overall objectives suited to each particular program. These overall objectives in turn are made more explicit and/or measurable through a number of general, terminal, and intermediate objectives.

In this Introduction to Computer Science Program, the educational aims are expressed by the following overall objectives:

(A) To develop the pupil's autonomy by introducing him to the thought processes which characterize computer science.

(B) To help the pupil to understand technological changes and the many applications of the computer in a constantly evolving world.

(A) To develop the pupil's autonomy by introducing him to the thought processes which characterize computer science.

These are some of the processes and activities which can help the pupil develop his autonomy:

— solving problems (in the broadest sense);

— collecting information;

— organizing information;

— communicating;

— creating;

— adapting his actions to reality (society, people, tools, etc.).

Designing and using tools (software) for information processing is an activity which involves all of these elements, one at a time or simultaneously. Indeed, writing and running a program involves:

1. **Solving a problem,** that is, using strategies and/or problem-solving heuristics.
2. **Collecting information** by applying problem-solving heuristics (heuristic method), by constructing programming algorithms, or by coding and debugging programs.
3. **Organizing information,** a basic activity in programming, which consists of organizing data (information to be processed), developing algorithms (step-by-step problem-solving procedures that can be carried out by the computer), and using the results.

---

1. Ministère de l'Éducation, *The Schools of Québec, Policy Statement and Plan of Action* (Québec, 1979), p.26

4. **Communicating:** This activity is carried out on several levels. The pupil communicates when he applies certain strategies and heuristics in designing or writing a computer program. He also uses a form of communication when he codes data and information, effectively handling the syntax and the semantics of a programming language. Finally, the communication process is at play in all interactions between teacher and pupil, every time the pupil explains his problems and difficulties to his teacher, for instance.

5. **Creating:** Writing a program is a creative activity. This activity is particularly relevant and rewarding as it may involve subjects which are directly related to the pupil's experience. Moreover, in the computer field, for any given problem there are frequently a great number of completely different "solutions" (programs, algorithms, processes). Even an imperfect solution always contains a number of good elements which may lead to a more complete solution. The pupil finds this very encouraging and acquires a taste for creative activities.

6. **Adapting to reality:** Intellectual independence should lead to action. It is important that the pupil adjust himself to reality, that is, to society, people, and the tools he uses. Constructing algorithms is both an intellectual and practical activity. There is a constant dialectical relationship between ideas, problem-solving structures, and their materialization into a computer program.

Learning how to program a computer and understanding the power and limits of the computer helps one to understand society in the computer age. The objectives pertaining to problem-solving, programming algorithms, coding, and machines, promote this type of knowledge and understanding. The subject of society is treated more explicitly in the objectives concerned with social impact and affective attitudes.

(B) To help the pupil to understand technological changes and the many applications of the computer in a constantly evolving world.

Man's ability to produce and to communicate is considerable. Quite early in his history, Man invented tools to supplement the strength of his muscles. It took him hundreds of thousands of years to develop communication tools and techniques (writing, printing, etc.) for broadening the range of his speech and recording his thoughts. He was later able to invent even more refined tools, such as devices for recording, reproducing, and retransmitting images and sounds at will.

Over the last ten years data processing machines have become more and more common. It is possible today to purchase a personal computer capable of solving all kinds of problems. This development may lead to a revolution as momentous as the invention of tools, writing, printing, and the steam engine.

Understanding technological changes involves more than simply learning to use a machine which expands man's thinking capabilities. It is also a question of mastering a new technology by learning to control the machine. In a introductory computer science course, learning how to program a computer is the first step toward achieving human control over the machine.

If he is to avoid becoming alienated by the machine, the pupil should be taught how to tame it, that is, to control it, to recognize its true value, and to understand the complex interactions between man, machines, and society. This critical attitude is an essential element in every citizen's education.

Among newly emerging technological changes, it is necessary to mention a few which are likely to be enduring human achievements.

Automation has affected the performance of many functions formerly executed by human beings, in particular in the manufacturing industries and the technical fields. In the future more and more manual work will be performed by robots equipped with micro-computers and micro-processors. It is important and urgent to consider the changes which such a situation will create.

Sophisticated calculators and computers will make work easier by taking over an increasing number of boring tasks in administration, accounting, research, and engineering, which require repetitive calculations. The consumer is about to witness the proliferation of micro-processors in the domestic and public sectors: home appliances, automobiles, measuring instruments, transportation services, banks, grocery stores, word processors, text editors, etc.

Also worth mentioning is the rapid growth of teleprocessing, a field which integrates telecommunications and information processing. Teleprocessing will soon encompass a large number of fields such as radio, television, telephone, data banks, education, information exchange, the press, etc.

These few examples show how essential it is to help the pupil understand current technological changes to prepare him adequately for the world of tomorrow.

Another Look at the Objectives of Education

It is clear that the two overall objectives described above are in agreement with the educational objectives of the schools, in particular those of the secondary school which are described in the POLICY STATEMENT AND PLAN OF ACTION:[1]

> To enable children and young people to develop according to their own talents and their own personal resources, to evolve into autonomous and creative individuals and to prepare themselves for their role as citizens. (p. 29, 2.3.1)
>
> To allow young people to pursue their general education and to orient themselves in life as individuals who are also members of a community. (p. 29, 2.3.2)
>
> The secondary school is concerned, for the most part, with preparation for professional and social life. (p. 30, 2.3.8)
>
> Adolescents grow up and, usually between the ages of 12 and 18 years, they become increasingly aware of the responsibilities which will become theirs when they have to decide on their future. The objectives of education in the secondary school take this fundamental characteristic of adolescence into account. (p. 30, 2.3.9)
>
> To introduce the pupils to the many areas of knowledge and technology in order to ensure their general education. (p. 31, 2.3.11)
>
> To instill working methods and an exact intellectual methodology, which can promote the progressive development of independent thought. (p. 31, 2.3.11)

1. Ministère de l'Éducation, op.cit.

To provoke in the students a type of learning conducive to integrating the acquisition of knowledge to personal experience. (p. 31, 2.3.11)

To bring the students to discover gradually the rewards and challenges of the society to which they belong. (p. 31, 2.3.11)

To consolidate and pursue in greater depth, by means of more systematic learning, the fields of knowledge and technology. (p. 31, 2.3.13)

To provoke personal commitments which emphasize the feeling of belonging and encourage participation and creativity. (p. 31, 2.3.13)

To work at increasing in young people the sense of individual and collective responsibility. (p. 31, 2.3.13)

To promote in students the structuring of coherent thought, through both optional and compulsory subjects. (p. 31, 2.3.13)

To develop in students a critical attitude in the face of the various intellectual trends by which they are sollicited. (p. 31, 2.3.13)

To help young people find a meaning to life. (p. 31, 2.3.13)

To facilitate access to future professional life. (p. 31, 2.3.13)

1.3 Continuity Between the Program and "Introduction to a Programming Language"

This program replaces the experimental course "Introduction to a Programming Language" (I.L.P. 110-441/541 for French-speaking pupils, C.S. 610-441/541 for English-speaking pupils). This particular course had been offered for several years to a relatively small number of pupils (1500 to 2000 pupils a year) under quite difficult conditions. As there was no official program designed by the ministère de l'Éducation, each school board had to submit its course outline and syllabus to the Ministère and request official permission for its implementation. In addition, the schools had to deal with a shortage of qualified personnel and a lack of materiel and financial resources.

The original objectives of the "Introduction to a Programming Language" course can no longer meet current needs and are still less suited to the needs of the future. It was necessary, therefore, to make available to the school system a first official introductory Computer Science Program designed for secondary school pupils.

This program will be of great benefit to those pupils who intend to pursue more specialized post-secondary studies and to enter a profession in the field of their choice. Finally, there is a strong possibility that computer science will be taught in the elementary schools in the near future.

1.4 Target Populations

This program is intended for Secondary IV and Secondary V pupils. Although it is designed for pupils in the general sector, it can also be used, under certain conditions, in the special education sector. The structural flexibility of the program should make it possible for all secondary school pupils to learn computer science. The program differs somewhat from other subjects in the extent to which it necessitates a multidisci-

plinary spirit. Normally, then, a computer science class should be as representative as possible of the school population, and include pupils of various academic backgrounds and interests.

1.5 Course Organization

The courses in the program, Introduction to Computer Science, are intended for Secondary IV and Secondary V pupils. The two courses, one for each year, are optional. Each course has been assigned four credits and should normally be covered in 100 hours. Both courses may be offered concurrently in a single year instead of consecutively over two years. However, it is preferable to offer one course per year.

The program is designed in such a way that a pupil who registers only for the first course (Secondary IV) and skips the second course (Secondary V) should still find it rewarding. Indeed, the first course is complete in itself and separate from the second one, even though it is a prerequisite to the latter.

1.6 Specific Conditions of Implementation

In order for the program, Introduction to Computer Science, to fully achieve all its goals and objectives, certain minimum conditions (instructional, material, etc.) must be met. For this reason, the ministère de l'Éducation will make appropriate guidelines for a controlled and coherent implementation available along with the program.

Consequently, all the school boards, including those who already offer the "Introduction to a Programming Language" course, should be able to implement the program, Introduction to Computer Science, in a context which fosters rational development.

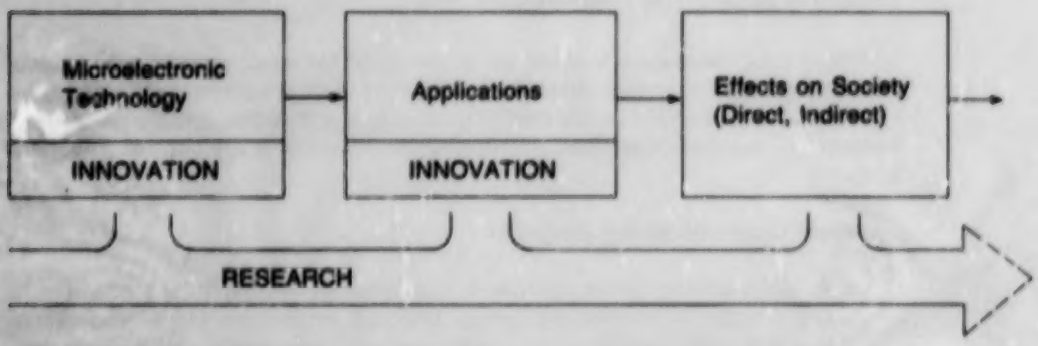
2
General Orientations of the Program

2. General Orientations of the Program

2.1 Socio-Cultural Dimension

The program, Introduction to Computer Science, is an outgrowth of the situation created by the explosive rise of microelectronic technology. This technology is responsible for a number of innovations whose direct and indirect impact on society is so strong and so varied that it is no longer possible to view the phenomenon as a routine development.

A simple diagram illustrates the process at work:



Wise, Chen, and Yokely have proposed a taxonomy which can help analyze the relationship between the applications and the effects of a given technology.[1] This system can be summed up by the simple listing of its five major categories:

- **Substantive Aspects:** resources, economy, society, technology, culture, psychology, security, internationality.
- **Basic Needs:** food, clothing, housing, transportation, health.
- **Institutions:** family, corporations, trade unions, schools, government.
- **Social Values:** community, justice, prosperity, equality, freedom, privacy.
- **Activities:** work, recreation, learning, continuing education.

Upon analyzing the elements of this brief classification, it becomes evident that our entire environment has been affected by the computer and information processing. We cannot ignore this fact, nor can we continue to pretend that our lives have not been directly or indirectly influenced by the computer. It is essential that we learn to control the machine if we are to avoid an unwelcome confrontation. The real danger lies in an attitude of submission to and dependence on the computer. Only through education and purposeful reflection will we be able to avoid these problems and to successfully integrate the computer into all human activities.

1. K.D. Wise, K. Chen, and R.E. Yokely, Microcomputers: A Technology Forecast and Assessment to the Year 2000 (Toronto: John Wiley & Sons, 1980).

Such are the concerns which underlie the program, Introduction to Computer Science. The courses outlined in this program should make it possible for the Secondary IV and Secondary V pupils to acquire a more complete general education, which should improve their ability to control their environment.

To achieve these objectives, the program has been presented as a prescriptive document, the nature of which requires specific actions on the part of the teacher. Although all the elements of the program are compulsory as they now stand, they can also be modified and adapted when necessary. The program itself remains open to future developments.

2.2 Psychoeducational Dimension

While it is not necessary to comment at length on the psychoeducational aspects of the program, it is useful, however, to highlight certain essential elements. The objectives and related content will thus be more readily understood, and the relationship between the situation described earlier and the topics of study will become more apparent.

(A) **Major Concerns of the Program:**

- Problem-Solving: Solving problems fosters the knowledge of the fundamentals of information processing (equipment, language, procedures, etc.). It also contributes to the development of the intellectual and psychological processes necessary for a better understanding of the world.
- These intellectual tools consist of the methods, techniques, and rules which the pupil learns, or the processes which he follows. While being conscious of the methods and processes he uses can certainly improve the pupil's efficiency, it is much less important that he be able to describe, identify, and list processes and methods. The point is to help the pupil acquire certain tools, and not to enable him to recite a whole catalogue of tools.
- For a given problem the combination of heuristic strategies and skills which are applied varies from one pupil to the next and depends on the way the individual functions. This should be taken into account in teaching strategies and evaluation. Besides, the list of objectives is virtually exhaustive and it is unlikely that it can be completed by any pupil. Consequently, what should be evaluated is not simply the number of objectives or the use of a particular method, but also the variety and effectiveness of the combination of strategies and heuristics used by each pupil.
- The development of algorithmic skills: These are indispensable for programming and information processing in general. These skills also find certain applications in other sectors of interest or in other disciplines.
- Handling the large amount of information available requires a relatively large number of activities, depending on the situation. These activities involve identifying information sources (data banks, magazines, journals, newspapers, libraries, bookstores), selecting, organizing, processing, or analyzing essential or useful data.
- Creating and constructing new tools for analyzing a problem or for developing one or several solutions.

- The nature of the program requires that consideration be given to the field to which its objectives and content apply and the context in which these are defined. This field integrates several areas of knowledge or sectors of activities, thus giving support to an interdisciplinary approach. It is important to realize that it is by establishing links with other disciplines that the pupil will be made aware of all the dimensions of computer science.

- These comments are meaningful only to the extent that the program is implemented in a dynamic manner and leads to the completion of concrete projects by the pupil. It is envisaged that the program will therefore not be taught using a passive approach.

- The individual is always the prime concern of the objectives, the related content, and the methodology. The hardware and software hold second place. This priority should be kept in mind when reading this program.

- It is not always easy to select the most effective and judicious approach to a given field without taking into account several relative factors. There are two schools of thought, and often their respective positions are likely to lead to the fragmentation of available resources and the division of the subjects of study. The first school of thought advocates that the teaching of computer science should focus on its applications to essentially practical situations. The second school of thought argues that only the theoretical aspects of computer science deserve to be taught.

  This program is designed to facilitate the merging of theory and practice. Its stated goals will be achieved only if these two aspects are balanced. In this approach, the theory and practice of computer science illuminate, explain, and complement each other.

It is essential to understand the entire program, and in particular the objectives, if its major orientations are to be followed. Computer science, as it is conceived here, is a special and relatively recent discipline. Its steady and rapid development as well as the dynamic and sometimes difficult use of the computer itself give rise to certain specific problems.

Thus, a number of unfamiliar terms will be found, while other known terms acquire a new meaning which belongs specifically to the computer field. Still other terms are part of the technical or scientific jargon (hence the short computer glossary in appendix).

Certain objectives refer specifically to certain intellectual processes which are considered important in this program. A particular methodology is recommended. For instance, Objective 2.20, TO **MODEL: to use the modelling process,** covers several categories of problems, the field of investigation being defined by the pupil, the teacher, and to a certain extent the general context of the school. This objective can take various meanings, from a graphic representation to the construction of a mathematical model, or a concrete representation (a model).

For instance, the modelling concept is found in the situation in which the pupil is asked:

— to find the curve which corresponds to the trajectory of a car on a race-course;

— to formulate the mathematical equation of that curve;

— to graph that curve on a cathode ray screen.

Although it is neither too convenient nor easy to formulate them in terms of terminal and intermediate behavioral objectives, these intellectual and learning processes are essential. They must therefore be included in the list of objectives, making allowances for a flexible yet clear interpretation by anyone who wishes simply to acquire some of the basic knowledge necessary for understanding computer science and its teaching.

(B) Instructional Approach

It is useful to recall some of the essential aspects of the instructional approach recommended for this program:

- The program emphasizes the individual, who is considered responsible for his own learning. Consequently, computer science is at the pupil's service, and not the other way around.
- Close and continuous attention should be paid to differences among individuals, particularly with respect to the ways in which they function.
- The pupil should be active and involved in concrete projects. The teacher should therefore closely supervise the pupil's progress. His interventions should suit the situations. The nature and quality of the relationships among the various people involved (other pupils, other teachers, other individuals, etc.) are therefore quite important.
- The interdisciplinary approach is strongly recommended. However, this approach requires that projects be selected on the basis of a number of strictly defined criteria. Indeed, to be able to solve a problem in another discipline, it is necessary to have a considerable knowledge of the characteristics of the discipline involved (concepts, rules laws, methods, etc.).
- Teaching and learning should be based on a specific pedagogical principle, namely, integrated instruction. Although from a logical point of view the development of tools should precede their applications, and organization of a task should come before its performance, educators know that "practice makes perfect." Consequently, a linear and sequential teaching of the objectives is not recommended. Sharper skills and a more enduring motivation can be developed by the pupil if pertinent concepts from the general objectives are studied in the context of a project or a problem. It is more profitable to take a problem as a starting point instead of studying each general objective before tackling related problems. This roundabout approach makes it possible to achieve several objectives simultaneously if the activities and projets are appropriately selected.
- Evaluation of these learning activities should be performed with particular care. The teacher has a crucial role to play in this evaluation process. The emphasis placed on intellectual processes, among others, does not decrease the need to establish frames of reference. Educational evaluation should keep its proper place and role.
- A proper balance between "closed" activities and "open" activities should be achieved. For an effective "open" approach (problems, algorithms, processes, etc.), it is essential that the pupil be adequately prepared by certain preliminary activities (closed approach). The teacher should analyze the entire situation so as to ensure that the preliminary activities are made available to the pupils at the right moment.
- The first three objectives, particularly General Objective 1 and General Objective 2, may seem redundant. It must be understood that human beings have imparted

to computer science a certain structure which reflects their own mental processes and functions. It is normal to find an objective or a concept which is applicable either to the functions of a human or to those of the computer. This dual perspective, which encompasses both human beings and the computer, highlights the need to have an overall understanding of the program in order to distinguish clearly the contexts of its implementation and to avoid confusion.

3.
Objectives

3. Objectives

The program includes six general objectives. The first and the third objectives concern the different jobs in the computer field, from those performed by the "project leader" and the "programming analyst" to those tasks executed by the "computer programmer" and the "computer operator." The second objective involves the acquisition and use of basic problem-solving skills by the pupil. These intellectual skills will be developed throughout the course. The fourth objective concerns knowledge of the hardware. The fifth objective pertains to the role of the computer and information processing in society, and the sixth objective focuses on the relationships between the pupil and computer science.

Even if there are a surprising number of objectives, they are all compulsory. However, the following factors should be taken into account:

- Depending on the languages and equipment used, only certain objectives are applicable.
- In any given problem, one usually deals with only certain objectives. To cover all the objectives it is necessary to handle a broad set of problems.
- Several objectives or sets of objectives are likely to be involved in a simple situation or operation.
- Although the number of objectives may seem excessive, their attainment is feasible in practice.

The program first lists only the general objectives (p. 21), then the general objectives along with the corresponding terminal objectives (p. 21), and finally all the general, terminal, and intermediate objectives (p. 24). Each intermediate objective is followed by a note (one or two numbers), which indicates in which grade this particular objective should be taught. For instance, "4" indicates a Secondary IV objective and "5" a Secondary V objective, whereas "4,5" specifies that a given objective should be taught in both Secondary IV and Secondary V.

3.1 General Objectives (*)

1. **TO CONSTRUCT programminig algorithms for solving problems related to everyday life situations.** 30%

2. **TO DEVELOP skills for solving problems by applying a broad variety of strategies and/or heuristics.** 30%

3. **TO PROGRAM a computer.** 30%

4. **TO KNOW the fundamental structures and components of a computer system, as well as the history of the computer.** 4%

5. **TO EVALUATE the place and the role of the computer in society.** 3%

6. **TO DEVELOP critical attitudes with respect to the computer and information processing.** 3%

3.2 General and Terminal Objectives

1. **TO CONSTRUCT programming algorithms for solving problems related to everyday life situations.**

   1.1 TO WRITE a clearly stated proposal for a project inspired by everyday life experiences;

   1.2 TO REFORMULATE the statement of the project selected;

   1.3 TO ORGANIZE the project using a process which makes it possible to go from the initial information to the desired output;

   1.4 TO DESCRIBE the process in a systematic way, taking into account the capabilities of the programming language and of the computer;

   1.5 TO TRANSLATE the process into a codable algorithm;

   1.6 TO CODE the different steps as programming statements;

   1.7 TO RUN the program on the computer;

   1.8 TO REVISE and REFINE the program (evaluate, test, debug);

   1.9 TO PRODUCE, for the benefit of the other pupils, a presentable and comprehensible report on the problem or project that has been completed (report, progress notes, log book,etc.).

(*) The percentages next to each objective are relative and should be taken as simple suggestions. For further information please see Section 5.2 (p.00).

2. **TO DEVELOP skills for solving problems by applying a broad variety of strategies and/or heuristics.**

   2.1 TO SELECT information pertaining to a problem which can be solved by programming;

   2.2 TO FIND a simpler problem which is a part of a given problem;

   2.3 TO FIND a problem which is isomorphic to the problem to be solved;

   2.4 TO GENERALIZE a problem;

   2.5 TO SUBDIVIDE a problem into smaller problems;

   2.6 TO APPLY pertinent and effective methods of analysis;

   2.7 TO FIND analogies;

   2.8 TO DEDUCE: to use the deductive process;

   2.9 TO DEFINE: to use the definition process;

   2.10 TO FORMALIZE: to use the formalization process;

   2.11 TO DESCRIBE a problem, its solution, or a program;

   2.12 TO DESTRUCTURE: to use the destructuring process;

   2.13 TO TEST the validity of an action or a program;

   2.14 TO CLASSIFY the elements of a problem or its solution;

   2.15 TO EXPLAIN the elements of a problem or its solution;

   2.16 TO PROCEED by trial and error;

   2.17 TO ITERATE: to use the iterative process;

   2.18 TO USE the recursive process;

   2.19 TO SELECT a subset of the preceding strategies;

   2.20 TO MODEL: to use the modelling process;

3. **TO PROGRAM A COMPUTER.**

   3.1 TO USE constants in a program;

   3.2 TO USE variables in a program;

   3.3 TO USE the different arithmetic, Boolean, and alphanumeric operators in a program;

3.4 TO CONSTRUCT arithmatic and alphanumeric expressions;

3.5 TO USE assignment instructions in a program;

3.6 TO USE display instructions in a program;

3.7 TO USE data entry instructions in a program;

3.8 TO USE conditional and unconditional branch instructions in a program;

3.9 TO USE the predefined functions of the language selected for a given program;

3.10 TO USE iterative and recursive structures in a program;

3.11 TO USE subprograms in a program;

3.12 TO USE files in a program;

3.13 TO USE certain basic system commands;

3.14 TO USE instructions to produce graphics; (**)

3.15 TO USE instructions to create music with the computer; (**)

3.16 TO USE instructions which enable the computer to control various external devices. (**)

4. TO KNOW the fundamental structures and components of a computer system, as well as the history of the computer.

4.1 TO DESCRIBE briefly how a computer works;

4.2 TO CLASSIFY computers on the basis of certain criteria;

4.3 TO IDENTIFY the main peripheral units;

4.4 TO USE to control commands of the computer and its main peripheral units;

4.5 TO IDENTIFY different types of software;

4.6 TO IDENTIFY the different types of codes used in a computer;

4.7 TO MASTER binary numbers and their derivatives;

4.8 TO KNOW the history of the computer.

5. TO EVALUATE the place and the role of the computer in society.

5.1 TO DESCRIBE briefly the more common applications of the computer in our society;

5.2 TO EXPLAIN why and how the computer is a tool at the service of mankind;

5.3 TO EVALUATE the impact of the computer on individual rights and personal freedom;

5.4 TO IDENTIFY some of the effects of the use of computer systems on the job market;

5.5 TO EVALUATE the possibility of entering a career in the computer field or in a related sector.

6. **TO DEVELOP critical attitudes with respect to the computer and information processing.**

6.1 TO EVALUATE the true nature of the computer and information processing;

6.2 TO REACT correctly vis-à-vis the computer and information processing;

6.3 TO EVALUATE certain activities related to information processing;

6.4 TO EVALUATE his place within a group during a programming activity;

6.5 TO DEVELOP self-confidence and self-esteem by planning and writing computer programs.

3.3 General,Terminal, and Intermediate Objectives

1. **TO CONSTRUCT programming algorithms for solving problems related to everyday life situations.**

This objective emphasizes the role of the programming or systems analyst, and focuses on the activities associated with the planning of a computer program. It highlights in particular the major steps which the pupil has to follow to complete his work. The development of a program involves the following steps:

— analyzing the problem or the project to be completed;

— planning a solution;

— coding and correcting errors on the computer;

— verifying and testing the program on the computer;

— loading and running the corrected program on the computer;

— evaluating the completed program or project and writing a final report to provide appropriate documentation.

Another concern of this objective is the acquisition and development of personal program planning techniques and methods. The objective also promotes the development of the skills required for analyzing and synthesizing, for communicating, organizing, coordinating, and evaluating. Finally, this objective offers the pupil the opportunity to learn teamwork and cooperation.

1.1 **TO WRITE a clearly stated proposal for a project inspired by everyday life experiences.**

1.1.1 TO LIST the subjects of interest of each pupil; (4,5)

1.1.2 TO DRAW UP a list of suggested projects with the other pupils; (4,5)

1.1.3 TO SELECT a project; (4,5)

1.1.4 TO COMMUNICATE his intentions to the teacher and the other pupils; (4,5)

1.1.5 TO OUTLINE the different stages of the project to be completed; (5)

1.1.6 TO MAKE suggestions for improving his classmates' projects; (5)

1.1.7 TO INDICATE which suggestions have been accepted; (5)

1.1.8 TO WRITE a first draft of the project. (5)

1.2 **TO REFORMULATE the statement of the project selected.**

1.2.1 TO FORMULATE hypotheses which are pertinent to the project; (5)

1.2.2 TO SPELL OUT the objectives to be achieved by processing the information (what should be obtained); (4)

1.2.3 TO INDICATE the relationships between the initial data and the final data; (4)

1.2.4 TO INDICATE the procedure for processing the information; (4)

1.2.5 TO INDICATE the factors to be considered: initial data, unknowns, restrictions, etc. (4)

1.3 **TO ORGANIZE the project using a process which makes it possible to go from the initial information to the desired output.**

1.3.1 TO IDENTIFY the tasks to be completed; (4)

1.3.2 TO DESCRIBE each of these tasks. (4)

1.3.3 TO IDENTIFY the operations which the computer will be instructed to perform, taking into account its capabilities and limits; (4,5)

1.3.4 TO DRAW UP a work plan; (5)

1.3.5 TO ESTABLISH a work schedule; (5)

1.3.6 TO SHARE the tasks if the project involves teamwork; (5)

1.3.7 TO LIST in logical order the actions which the computer will be instructed to perform. (4)

1.4 TO DESCRIBE the process in a systematic way, taking into account the capabilities of the programming language and of the computer.

1.4.1 TO DEFINE the variables of the problem; (4)

1.4.2 TO INDICATE the parameters of the problem; (4,5)

1.4.3 TO IDENTIFY the relationships among the parameters; (4,5)

1.4.4 TO LIST in logical order the basic elements of the algorithm; (4)

1.4.5 TO INDICATE the procedures; (4,5)

1.4.6 TO INDICATE the branches; (4)

1.4.7 TO DIAGRAM the logical development of the different stages of the problem or project; (4,5)

1.4.8 TO DESCRIBE the procedure in pseudo-code in the pupil's mother tongue; (4,5)

1.4.9 TO DESCRIBE the structure of the model. (5)

1.5 TO TRANSLATE the process into a codable algorithm.

1.5.1 TO TRANSLATE certain stages of the algorithm into instructions for: reading, writing; assignment; comparison; arithmetic; alphanumeric, or logical operations; functions, procedures; etc; (4)

1.5.2 TO TRANSLATE the algorithm or part of the algorithm into instructions for a sequence of actions; (4)

1.5.3 TO TRANSLATE the algorithm or part of the algorithm into decision instructions: conditional decisions, alternative decisions, multiple decisions; (4)

1.5.4 TO TRANSLATE the algorithm or part of the algorithm into an iteration; (4)

1.5.5 TO TRANSLATE the algorithm or part of the algorithm into a recursive procedure. (5)

1.6 TO CODE the different steps as programming instructions (See General Objective 3). (4,5)

1.7 TO RUN the program on the computer.

1.7.1 TO LOAD the program into the main memory; (4)

1.7.2 TO SET the execution of the program in motion. (4)

1.8 TO REVISE and REFINE the program (evaluate, test, debug).

1.8.1 TO CORRECT syntax errors; (4)

1.8.2 TO CORRECT semantic errors; (4)

1.8.3 TO CORRECT errors in logic; (4)

1.8.4 TO TEST or TO SIMULATE the program using sufficient pertinent data; (4)

1.8.5 TO START the step-by-step execution of the program; (4,5)

1.8.6 TO PRINT OUT a record of the variables; (5)

1.8.7 TO DUMP the memory; (5)

1.8.8 TO INTERPRET error messages; (4)

1.8.9 TO CORRECT the output format (pictures, texts, numbers, music, etc.); (5)

1.8.10 TO INSERT new instructions; (4)

1.8.11 TO REMOVE certain instructions; (4)

1.8.12 TO MODIFY certain instructions; (4)

1.8.13 TO TEST the program again; (4)

1.8.14 TO STOP the program's execution in mid-course; (4)

1.8.15 TO IDENTIFY or TO LIST the parameters; (4)

1.8.16 TO VARY the parameters separately; (4)

1.8.17 TO RECORD the results; (4)

1.8.18 TO IDENTIFY areas where improvements are needed; (4)

1.8.19 TO INDICATE the differences between the model run on the computer and reality. (4)

1.9 **TO PRODUCE, for the benefit of the other pupils, a presentable and comprehensible report on the problem or project that has been completed (report, progress notes, log book, etc.).**

1.9.1 TO PRESENT an analysis of the problem; (4)

1.9.2 TO PRESENT the computer program; (4)

1.9.3 TO PRESENT a sample of the program output; (4)

1.9.4 TO WRITE personal comments about: (5)

— the completed project,
— the difficulties encountered,
— possible improvements;

1.9.5 TO EVALUATE the following aspects: (5)

— the final product,
— the validity of the model,
— the level of difficulty,
— the knowledge of the language acquired by the pupil,
— the amount of interest inspired in his classmates;

1.9.6 TO EXPLAIN the following aspects: (4)

— what is liked most,
— what is liked least,
— what is the easiest,
— what is the most difficult,
— what is the most useful,
— the contribution of the computer to the realization of a project,
— communication and cooperation with other classmates and the teacher;

1.9.7 TO NOTE the restrictions due to: (4,5)

— the pupil's insufficient knowledge,
— the limited amount of time available for completing a suitable project,
— the limitations of the computer
— other identifiable factors;

1.9.8 TO GENERATE original ideas. (4,5)

2. **TO DEVELOP skills for solving problems by applying a broad variety of strategies and/or heuristics.**

This general objective and the related terminal objectives aim to help the pupil to acquire the intellectual tools required for solving problems. It should be realized that planning, writing, and running a program require general problem-solving skills.

These activities also offer the opportunity to apply a broad variety of strategies and heuristics. Section 2.2, "Psychoeducational Dimension," a subdivision of the General Orientation part of this program, should be read again carefully, as it deals specifically with General Objective 2 and its related objectives.

2.1 TO COLLECT information pertaining to a problem which can be solved by programming.

- 2.1.1 TO FIND an exceptional case; (4)
- 2.1.2 TO FIND a basic case; (4)
- 2.1.3 TO FIND a simple case; (4)
- 2.1.4 TO FIND a limiting case; (4)
- 2.1.5 TO FIND a clue; (4)
- 2.1.6 TO FIND a residue (what is not treated) (4)
- 2.1.7 TO FIND a paradoxical case; (5)
- 2.1.8 TO FIND a concrete example; (4)

2.2 TO FIND a simpler problem which is a part of a given problem.

- 2.2.1 TO DEFINE a portion of a problem whose solution is self-evident; (5)
- 2.2.2 TO FIND a solution to a problem which is a portion of the principal problem. (5)

2.3 TO FIND a problem which is isomorphic to the problem to be solved.

- 2.3.1 TO FIND a problem which has been solved and which has an equivalent or identical structure. (5)

2.4 TO GENERALIZE a problem.

- 2.4.1 TO FIND a more general case; (5)
- 2.4.2 TO IDENTIFY certain characteristics of the problem; (5)
- 2.4.3 TO FIND certain rules which make it possible to go from one state to another; (5)
- 2.4.4 TO FIND a formula. (4)

2.5 TO SUBDIVIDE a problem into smaller problems.

- 2.5.1 TO BREAK the problem down into smaller sections; (4)

2.5.2 TO SUBDIVIDE the problem into a sequence of complementary smaller problems; (4)

2.5.3 TO SUBDIVIDE the problem into mutually exclusive smaller problems; (4)

2.5.4 TO SUBDIVIDE the problem into overlapping smaller problems; (5)

2.6 TO APPLY pertinent and effective methods of analysis.

2.6.1 TO APPLY the top-down method of analysis (from the abstract to the concrete); (5)

2.6.2 TO APPLY the bottom-up method of analysis (from the concrete to the abstract); (5)

2.6.3 TO APPLY the progressive method of analysis (from the principles to the consequences); (5)

2.6.4 TO APPLY the regressive method of analysis (from the consequences to the principles); (5)

2.7 TO FIND analogies.

2.7.1 TO FIND a model; (5)

2.7.2 TO ASCERTAIN whether a model corresponds to reality; (5)

2.7.3 TO ASCERTAIN whether a model does justice to reality; (5)

2.7.4 TO DETERMINE whether a particular model is inaccurate or unrealistic. (5)

2.8 TO DEDUCE: to use the deductive process.

2.8.1 TO APPLY a rule which makes it possible to go directly from the present situation to the desired situation; (4,5)

2.8.2 TO APPLY a set of intermediate rules which make it possible to go from the present situation to the desired situation; (4,5)

2.8.3 TO VERIFY whether a demonstration is accurate. (4,5)

2.9 TO DEFINE: to use the definition process.

2.9.1 TO COMPOSE a functional definition; (5)

2.9.2 TO COMPOSE a structural or analytical definition; (5)

2.10 TO FORMALIZE: to use the formalization process.

2.10.1 TO IDENTIFY certain formal aspects; (5)

2.10.2 TO ABSTRACT a number of specific aspects; (5)

2.10.3 TO SYMBOLIZE certain elements of the solution (4,5)

2.11 TO DESCRIBE a problem, its solution, or a program.

2.11.1 TO DESCRIBE a problem, a solution, or an action, in everyday language; (4)

2.11.2 TO DESCRIBE the solution of a problem in a pseudo-code in the pupil's mother tongue; (4)

2.11.3 TO DESCRIBE the sequential structure of a program or of an action; (4)

2.11.4 TO DRAW diagrams (graphs, flowcharts, structure diagrams, etc.). (4)

2.12 TO DESTRUCTURE: to use the destructuring process.

2.12.1 TO RECOGNIZE the component parts of a given structure; (4)

2.12.2 TO RE-EXAMINE these component parts in order to make needed corrections. (4)

2.13 TO TEST the validity of an action or a program.

2.13.1 TO SIMULATE the performance of an action or the execution of a program by playing roles; (4)

2.13.2 TO DESCRIBE each of the different steps of a program (4)

2.13.3 TO SIMULATE the execution of a program; (4)

2.13.4 TO OBTAIN a record of the variables of a program at each major stage. (4)

2.14 TO CLASSIFY the elements of a problem or its solution.

2.14.1 TO IDENTIFY relations; (4)

2.14.2 TO GROUP elements which share common characteristics; (4)

2.14.3 TO ORDER elements; (4,5)

2.14.4 TO CREATE subsets; (5)

2.14.5 TO IDENTIFY invariants; (4,5)

2.14.6 TO ASSOCIATE different elements with one another; (4)

2.14.7 TO COMPARE different elements. (4)

2.15 TO GIVE an explanation of the elements of a problem or its solution.

2.15.1 TO WRITE a problem or a solution in minute detail; (4)

2.15.2 TO VERBALIZE a problem or a solution; (4)

2.15.3 TO DESCRIBE a problem, a solution, or an action for the benefit of others; (4)

2.15.4 TO FURNISH the pertinent information; (4)

2.15.5 TO CONSULT various sources to obtain information on a given subject. (4,5)

2.16 TO PROCEED by trial and error.

2.16.1 TO VERIFY a given statement by the substitution of randomly chosen values; (4)

2.16.2 TO VERIFY a solution by the substitution of randomly chosen values; (4)

2.16.3 TO VERIFY a formula by the substitution of randomly chosen values; (4)

2.16.4 TO FIND the correct solution of a problem by substituting values into formulas. (4)

2.17 TO ITERATE: to use the iterative process.

2.17.1 TO REPEAT an action which has proven effective; (4)

2.17.2 TO DISCOVER a stop rule; (4)

2.17.3 TO DEBUG a program in successive steps. (4)

2.18 TO USE the recursive process.

2.18.1 TO USE recursion; (5)

2.18.2 TO USE the same process within a given process. (5)

2.19 TO SELECT a subset of the preceding strategies. (5)

2.20 TO MODEL: to use the modelling process.

2.20.1 TO MAKE a simplified representation; (4)

2.20.2 TO MAKE a schematic representation; (4)

2.20.3 TO CONSTRUCT an arbitrary model which looks externally the same as a given problem; (4)

2.20.4 TO MAKE a structured representation; (4)

2.20.5 TO CONSTRUCT a general model; (5)

2.20.6 TO MODIFY a model. (5)

3. TO PROGRAM a computer.

The following terminal objectives are suitable for most programming languages. However, it is the teacher's responsibility to determine which of the intermediate objectives are appropriate for the language and the equipment he uses in his class. It must also be pointed out that TO PROGRAM means, in this context, to code the programming algorithms, to put the program in functioning order, to evaluate the results, and to correct the program.

3.1 TO USE constants in a program.

3.1.1 TO USE numeric constants; (4)

3.1.2 TO REPRESENT constants expressed in scientific notation in decimal notation, and vice-versa; (4)

3.1.3 TO IDENTIFY the degree of precision of the numeric constants; (4)

3.1.4 TO USE alphanumeric constants; (4)

3.1.5 TO USE double precision numeric constants; (5)

3.1.6 TO DIFFERENTIATE between rational and integer constants; (5)

3.1.7 TO USE constants of type list. (4,5)

3.2 TO USE variables in a program.

3.2.1 TO IDENTIFY different types of variables: numeric, subscripted, list, alphanumeric, logical; (4)

3.2.2 TO USE integer numeric variables; (4)

3.2.3 TO DIFFERENTIATE between rational and integer variables; (5)

3.2.4 TO USE subscripted variables; (5)

3.2.5 TO USE variables of type list; (4,5)

3.2.6 TO USE alphanumeric variables; (4)

3.2.7 TO USE logical variables. (5)

3.3 TO USE the different arithmetic, Boolean, and alphanumeric operators in a program.

3.3.1 TO IDENTIFY the unary, binary, and function operators in the language; (4)

3.3.2 TO IDENTIFY the symbols used for addition, subtraction, multiplication, division, and exponentiation; (4)

3.3.3 TO APPLY the rules of priority with respect to the order of operations; (4)

3.3.4 TO CHANGE the order in which the operations are performed, using parentheses; (4)

3.3.5 TO USE logical operators: conjunction, disjunction, and negation; (5)

3.3.6 TO USE alphanumeric string operators; (4)

3.3.7 TO USE list operators. (4,5)

3.4 TO CONSTRUCT arithmetic and alphanumeric expressions.

3.4.1 TO TRANSLATE algebraic expressions into expressions which conform to the characteristics of the language used; (4)

3.4.2 TO CORRECT the errors in a given expression; (4)

3.4.3 TO TRANSLATE a set of operations on alphanumeric strings and/or lists into expressions which conform to the characteristics of the language used; (4,5)

3.5 TO USE assignment instructions in a program.

3.5.1 TO INITIALIZE the variables; (4)

3.5.2 TO USE assignment instructions with arithmetic or alphanumeric expressions; (4)

3.5.3 TO USE assignment instructions, using an incrementing or decrementing counter; (4)

3.5.4 TO USE assignment instructions which allow accumulation of values. (4)

3.6 TO USE display instructions in a program.

3.6.1 TO INSTRUCT the computer to display numbers without format specifications; (4)

3.6.2 TO INSTRUCT the computer to display alphanumeric strings without format specifications; (4)

3.6.3 TO INSTRUCT the computer to display numbers according to a specific output format; (5)

3.6.4 TO INSTRUCT the computer to display alphanumeric strings according to a specific output format; (5)

3.6.5 TO DIRECT the desired information toward various output devices. (5)

3.7 TO USE data entry instructions in a program.

3.7.1 TO INPUT data using the keyboard; (4)

3.7.2 TO INSTRUCT the computer to read the data included in the program itself; (4)

3.7.3 TO INSTRUCT the computer to read the data in a file; (5)

3.7.4 TO INPUT data using other available peripherals such as the card reader, magnetic tapes, punched paper tapes, disks, the joystick, the analog/digital converter, etc... (4,5)

3.8 TO USE conditional and unconditional branch instructions in a program.

3.8.1 TO IDENTIFY branch instructions; (4)

3.8.2 TO CHANGE the sequence of execution of a program, using branch instructions; (4)

3.8.3 TO USE relation operators and/or logical operators in the conditional expressions; (5)

3.8.4 TO USE conditional branch instructions with two or three branches to modify the logical sequence of execution of a program; (4)

3.8.5 TO USE branch instructions with n branches to solve decision problems; (5)

3.9 TO USE the predefined functions of the language selected for a given program.

3.9.1 TO USE predefined arithmetic functions in a program; (4)

3.9.2 TO IDENTIFY potential programming errors due to restrictions in the arguments of certain functions; (4)

3.9.3 TO USE predefined alphanumeric functions in a program; (4)

3.9.4 TO USE predefined utility functions in a program; (4)

3.9.5 TO USE the appropriate programming structures to avoid errors in the use of predefined functions. (4)

3.10 TO USE iterative and recursive structures in a program.

3.10.1 TO USE iterative structures of the type "Repeat....until;" (4)

3.10.2 TO USE iterative structures of the type "As long as;" (4)

3.10.3 TO USE a counter to control an iterative process (in particular to exit from a loop); (4)

3.10.4 TO USE a stop rule to end an iterative process; (4)

3.10.5 TO USE nested iterative structures in a program; (4)

3.10.6 TO USE recursive structures in a program; (4,5)

3.11 TO USE subprograms in a program.

3.11.1 TO IDENTIFY the role of parameters in subprograms; (5)

3.11.2 TO DISTINGUISH between local variables and global variables in subprograms; (5)

3.11.3 TO USE function subprograms; (5)

3.11.4 TO USE subroutine subprograms; (5)

3.11.5 TO USE procedure subprograms; (4,5)

3.11.6 TO DESCRIBE the similarities and differences between the various types of subprograms. (5)

3.12 TO USE files in a program.

3.12.1 TO SHOW how to create, modify, delete, read from, and write to, files; (5)

3.12.2 TO IDENTIFY the various types of files: program files, random access files, sequential access files; (5)

3.12.3 TO USE program file commands; (4)

3.12.4 TO USE random and sequential access data file instructions; (5)

3.12.5 TO CALCULATE the length (the space) required for a given file; (5)

3.12.6 TO TEST for an end-of-file. (5)

3.13 TO USE certain basic system commands.

3.13.1 TO LIST a program; (4)

3.13.2 TO EDIT a program or a file; (4)

3.13.3 TO INSTRUCT the computer to execute a program; (4)

3.13.4 TO RECALL a stored program to the work space; (4)

3.13.5 TO ERASE a program; (4)

3.13.6 TO OBTAIN a catalogue of the files kept on cassettes or disks in the system; (4)

3.13.7 TO STORE a program in the auxiliary memory; (4)

3.13.8 TO STOP the execution of a program. (4)

3.14 TO USE instructions to produce graphics. (**)

3.14.1 TO PRODUCE graphic designs; (4,5)

3.14.2 TO USE graphic an[illegible]ation techniques; (4,5)

3.14.3 TO DRAW the graphs of mathematics functions; (4,5)

3.14.4 TO DRAW statistical graphs: bar graphs, line graphs, histograms, picture graphs, circular diagrams, etc. (4,5)

3.15 TO USE instructions to create music with a computer. (**)

3.15.1 TO USE instructions to create music; (4,5)

3.15.2 TO USE instructions to edit music; (4,5)

3.15.3 TO USE instructions to control musical instruments and devices. (4,5)

3.16 TO USE instructions which enable the computer to control various external devices. (**)

3.16.1 TO USE instructions to encode or decode digital/analog data; (4,5)

3.16.2 TO USE instructions which enable the computer to turn on various external devices (motors, switches, alarm systems, etc...); (4,5)

3.16.3 TO USE instructions to turn off various external devices. (4,5)

(**) Depending on available resources.

4. TO KNOW the fundamental structures and components of a computer system as well as the history of the computer.

4.1 TO DESCRIBE briefly how a computer works.

4.1.1 TO IDENTIFY the main components of a computer: the central processing unit, the peripherals; (4)

4.1.2 TO IDENTIFY the basic operations performed by a computer: input, processing, output; (4)

4.1.3 TO DESCRIBE briefly the functions of the main components of a computer. (4)

4.2 TO CLASSIFY computers on the basis of certain criteria.

4.2.1 TO CLASSIFY computers according to their size: microcomputers, minicomputers, mainframe computers; (4)

4.2.2 TO CLASSIFY computers according to the different ways of processing information: batch processing, real-time processing, time sharing; (5)

4.2.3 TO DESCRIBE briefly the set-up of a network of computers; (5)

4.2.4 TO DESCRIBE briefly the different categories of computers. (5)

4.3 TO IDENTIFY the main peripheral units.

4.3.1 TO IDENTIFY the different categories of peripheral units: input units, output units, auxiliary memory units; (4)

4.3.2 TO IDENTIFY and TO DESCRIBE different input units: the alphanumeric keyboard, the card reader, the punched tape reader, the magnetic tape reader, etc...; (4)

4.3.3 TO IDENTIFY and TO DESCRIBE briefly different output units: the video display screen, the printer, the plotter, etc...; (4)

4.3.4 TO IDENTIFY and TO DESCRIBE briefly different auxiliary memory units: the magnetic disk unit, the magnetic tape unit, the bubble memory unit. (4)

4.4 TO USE the control commands of the computer and its main peripheral units.

4.4.1 TO CONNECT the computer and the peripherals as required; (4)

4.4.2 TO USE the control commands for the central processing unit of the computer; (4)

4.4.3 TO USE the control commands for the keyboard; (4)

4.4.4 TO USE the control commands for the video display screen; (4)

4.4.5 TO USE the control commands for the reader; (4)

4.4.6 TO USE the control commands for the printer; (4)

4.4.7 TO USE the control commands for the plotter; (4)

4.4.8 TO USE the control commands for the memory unit; (4)

4.4.9 TO USE the control commands which regulate communication between the computer and the peripheral units. (4)

4.5 TO IDENTIFY different types of software.

4.5.1 TO DISTINGUISH between a high-level language and a low-level language; (4)

4.5.2 TO DEFINE machine language; (5)

4.5.3 TO DEFINE an assembler; (5)

4.5.4 TO DEFINE a compiler; (5)

4.5.5 TO DEFINE an interpreter; (5)

4.5.6 TO DEFINE an operating system; (5)

4.5.7 TO DISTINGUISH between hardware and software. (4)

4.6 TO IDENTIFY the different codes used in a computer.

4.6.1 TO DEFINE numeric code; (5)

4.6.2 TO RECOGNIZE a decimal number in its binary code; (5)

4.6.3 TO DEFINE ASCII code. (5)

4.7 TO MASTER binary numbers and their derivatives.

4.7.1 TO CONVERT a decimal number to binary; (5)

4.7.2 TO CONVERT a binary number to decimal; (5)

4.7.3 TO ADD two binary numbers; (5)

4.7.4 TO SUBTRACT two binary numbers; (5)

4.7.5 TO MULTIPLY two binary numbers; (5)

4.7.6 TO DIVIDE a binary number by another; (5)

4.7.7 TO CONVERT a binary number to an octal number; (5)

4.7.8 TO CONVERT an octal number to a binary number; (5)

4.7.9 TO CONVERT a binary number to an hexadecimal number; (5)

4.7.10 TO CONVERT a hexadecimal number to a binary number. (5)

4.8 TO KNOW the history of the computer.

4.8.1 TO DESCRIBE the beginnings of the computer era; (4)

4.8.2 TO DESCRIBE the roles played by the first inventors of the computer. (4)

4.8.3 TO IDENTIFY the major stages in the development of the computer. (4)

5. TO EVALUATE the place and the role of the computer in society.

The aim of this objective is to enable the pupil to understand the place of the computer in society as well as its roles and functions. The objective can be attained through the knowledge of certain facts, concepts, and ideas transmitted to the pupil by the teacher (and the media he uses), as well as through the pupil's own reflection on his experience with the computer. Consequently, this objective does not require active and sustained involvement on the part of the teacher. It is the pupil, rather, who should have certain experiences and think about them. That is in effect what happens when he learns to program and to use a computer.

5.1 TO DESCRIBE briefly the more common applications of the computer in our society.

5.1.1 TO DESCRIBE briefly the role of the computer in business management; (4)

5.1.2 TO DESCRIBE briefly the role of the computer in communications; (4)

5.1.3 TO DESCRIBE briefly the role of the computer in the use of data banks; (4)

5.1.4 TO DESCRIBE briefly the role of the computer in industrial process management; (4)

5.1.5 TO DESCRIBE briefly the role of the computer in the educational system; (4)

5.1.6 TO DESCRIBE briefly the role of the computer in research and development; (4)

5.1.7 TO DESCRIBE briefly the role and the uses of the personal computer. (4)

5.2 TO EXPLAIN why and how the computer is a tool at the service of mankind.

5.2.1 TO UNDERSTAND that it is man who controls the machine; (4)

5.2.2 TO UNDERSTAND that most of the errors attributed to machines are in fact human errors; (4)

5.2.3 TO IDENTIFY programming problems which are difficult and complex, and sometimes open-ended; (4)

5.2.4 TO UNDERSTAND the implication of the phrase "The computer is intelligent;" (5)

5.2.5 TO EXPLAIN how the computer expands man's intellectual capabilities (memory capacity, speed in performing operations, large number of relationships it can discover); (5)

5.2.6 TO DETERMINE how the computer can improve the quality of and increase the quantity of information. (4)

5.3 TO EVALUATE the impact of the computer on individual rights and personal freedom.

5.3.1 TO DESCRIBE how the computer increases or restricts the freedom of individual citizens; (4,5)

5.3.2 TO UNDERSTAND that data banks may contain errors; (4)

5.3.3 TO IDENTIFY certain recourses available to citizens who wish to obtain the information about themselves contained in data banks, and to have that information corrected; (4,5)

5.3.4 TO EXPLAIN how the computer can enhance or hamper the independence of individuals (processing information themselves, more effective management, data banks, etc...); (4,5)

5.3.5 TO RECOGNIZE that problems pertaining to individual rights and personal freedom depend on the use made of data banks and not on these data banks themselves or on their computerization; (4,5)

5.3.6 TO RECOGNIZE that learning to use the computer can enhance the freedom of the competent user. (4)

5.4 TO IDENTIFY some of the effects of the use of computer systems on the job market.

5.4.1 TO IDENTIFY some job categories created by the development of the computer and information processing; (4)

5.4.2 TO IDENTIFY some job categories which have disappeared as a result of the development of the computer and information processing; (4)

5.4.3 TO IDENTIFY the fields where the applications of the computer can increase productivity. (4)

5.5 TO EVALUATE the possibility of entering a career in the computer field or in a related sector.

5.5.1 TO DESCRIBE briefly the major careers in the computer field and in related sectors; (5)

5.5.2 TO EVALUATE his preferences with respect to the different careers in the computer field; (5)

5.5.3 TO EVALUATE his aptitudes with respect to those careers; (5)

6. TO DEVELOP critical attitudes with respect to the computer and information processing.

These objectives are essential within the framework of a general education in secondary school. They enable the pupil who has mastered them to better understand modern society. They may even motivate him to go beyond this introduction to computer science either by pursuing a career in this discipline or by using intelligently the means made available by the computer.

Attainment of these objectives can be ensured particularly by a positive attitude on the part of the teacher, by timely and stimulating interventions on his part, and finally by a teaching methodology centered around the pupils' projects. This methodology requires that the teacher deal with programming projects selected by the pupils themselves instead of imposing on them exercises in which they may not be interested.

The program allocates only a small percentage of the total time for this general objective. It would indeed be contrived, inefficient, and even harmful, to devote entire periods to it. It is preferable that the teacher discretely integrate these objectives into his methodology and instructional activities.

6.1 TO EVALUATE the true nature of the computer and information processing.

6.1.1 The pupil should REALIZE that the computer and information processing are not a threat; (4,5)

6.1.2 The pupil should REALIZE that the computer can serve him. (4,5)

6.2 TO REACT correctly vis-à-vis the computer and information processing.

6.2.1 The pupil should PERCEIVE the computer as a tool at the service of human intelligence; (4,5)

6.2.2 The pupil should DEVELOP an understanding of technological progress and its effects; (4,5)

6.2.3 The pupil should REALIZE that one of man's characteristics is his ability to construct tools. (4,5)

6.3 TO EVALUATE certain activities related to information processing.

6.3.1 The pupil should REALIZE that the computer is a great technological achievement; (4,5)

6.3.2 The pupil should PERCEIVE programming as a problem-solving activity which can be very challenging at times; (4,5)

6.3.3 The pupil should UNDERSTAND that most work can be improved; (4,5)

6.3.4 The pupil should LEARN TO ENJOY refining his work; (4,5)

6.3.5 The pupil should REALIZE that programming is a creative activity; (4,5)

6.3.6 The pupil should DEVELOP greater interest in programming. (4,5)

6.4 TO EVALUATE his place within a group during a programming activity.

6.4.1 The pupil should LEARN to communicate with his classmates and his teachers; (4,5)

6.4.2 The pupil should LEARN to collaborate and cooperate with others; (4,5)

6.4.3 The pupil should REALIZE the impact of the computer on society. (4,5)

6.5 TO DEVELOP self-confidence and self-esteem by planning and writing computer programs.

6.5.1 The pupil should REALIZE that he can indeed program a computer; (4,5)

6.5.2 The pupil should EVALUATE his programming skills; (4,5)

6.5.3 The pupil should REALIZE that he can express himself through programming; (4,5)

6.5.4 The pupil should REALIZE that he is involved in a creative activity when planning, writing, and running a program. (4,5)

4
Evaluation

4. Evaluation[1]

4.1 Functions of Evaluation

Evaluation performs several functions and serves several purposes in education. It is used in particular to:

1. make a diagnosis;
2. modify a program of studies or a curriculum;
3. compare;
4. define needs;
5. determine whether objectives (and goals) have been attained.

1. **Making a diagnosis** usually involves detecting and identifying the reasons behind a pupil's learning difficulties with a view to making pertinent and effective instructional decisions. Appropriate tests and examinations are given for that purpose. The teacher, who has observed the pupil and examined his work, is the person best qualified to act in this context.

   An interesting and effective approach is to rely on the dialectical process by which the teacher and the pupil, through methodical interaction, assess the situation and take appropriate action.

2. **Program modification** is more specifically the concern of formative evaluation. It encompasses both the development of the program and its implementation in the classroom. The process requires that the content of the program be regularly checked against actual conditions. This process should guard against the unrestricted use of evaluation for classifying pupils. It helps to improve the quality of education by ensuring the continuous revision of the materials used in the program.

3. Evaluation is also useful for **comparing** programs, teaching methods, school organization, and other aspects of the educational system such as the material conditions. If educational choices are to foster a better education, they should naturally be evaluated. As these choices are based on knowledge and values, their validity should be tested by experience. Such a practice, if faithfully followed, should ensure the effectiveness of the educational process.

   Although comparative evaluation is complex and demanding, as it takes place over a long period of time and requires specific methodologies, it should play a central role in the educational process. It contributes to the improvement of the network of hypotheses and conclusions which underlies the educational system, thus making it possible for the educational system itself to improve.

4. **Defining needs** involves determining the timeliness and appropriateness of actions, policies, curriculum contents, etc.. This requires more than conducting interviews and polls because the needs expressed under those circumstances often reflect personal values and subjective views.

---

(1) This section is largely based on a work by Elliot Eisner (see the bibliography).

It is important to evaluate the relationships between these values and views, on the one hand, and the methods, tools, and needs selected, on the other hand. In this type of evaluation, it is essential that all the individuals involved integrate these elements.

5. **Determining whether the objectives have been attained** is another function of evaluation. The specificity of objectives is emphasized because they serve as criteria for evaluating the pupil's performance. Objectives are to a certain extent an indication of the effectiveness of the program. They should not, however, become boundaries. Formulating specific objectives is not an essential condition of evaluation.

   Evaluation is a means of collecting and analyzing information. The context in which this is done is such that everyone involved, including teachers, should of necessity perform this function.

4.2 Subjects of Evaluation

A program of studies designed to meet certain needs naturally makes use of one or several of these functions of evaluation. However, in the specific case of the Computer Science Program, three functions of evaluation are emphasized, and they apply specifically to the program itself, the teaching, and the learning outcomes.

1. **The program:** Evaluation of the program (activities, criteria, values, etc.) is essential and should take into account the different contexts of its implementation. It is necessary to know whether the content, the objectives (depending on the case), and the activities of the program are suited to the pupils' capabilities and correspond to their level of development and maturity.

   Beyond these three criteria (capabilities, development, and maturity), the evaluator should verify whether the presentation of the program has been adequately organized (instructional activities in sufficient number and variety, presentation of subjects using various media, diversity of the instructional approaches, etc.).

2. **The teaching:** Evaluation of the quality of teaching raises very complex problems. Knowledge in this field is still incomplete and the methodological difficulties are numerous. Besides the usual quantitative methods, new methods for assessing the qualitative aspects of teaching should be found.

3. **The learning outcomes:** Assessing the learning outcomes of a program (curriculum) during or at the end of the course is a characteristic concern of evaluation. This is also the aspect of the process to which the public is particularly sensitive.

   The outcomes also involve the objectives in the present context. An objective describes a specific point of arrival which may not always coincide with the actual result. Instead of considering a single point of arrival, it seems more worthwhile to consider several possibilities, since the means applied may lead to several different results. It is essential, therefore, to keep an open mind when evaluating learning outcomes.

Conclusion

In view of these considerations, it is advisable to encourage not one particular type of evaluation but rather the establishment of what Elliot Eisner calls an "evaluative

landscape," a much more effective process of evaluation which involves numerous critical discussions.

4.3 Recommendations

Considering the issues it raises, the values on which it is based, the objectives it pursues, and the subject matter it involves, the Computer Science Program requires a specific and pertinent type of evaluation.

With respect to the program, various types of evaluation, including in particular linear evaluation, should be conducted in order to determine the appropriateness of the items selected (content,activities, orientations, etc.), taking into account such elements as the context in which the program is implemented, the level of development and maturity of the pupils, etc.

With respect to the teaching, it is advisable to design and to use tools which will ensure a quality of teaching that is in tune with prevailing conditions and with the more progressive developments in the field. These tools, some of which have yet to be developed, should make it possible to carry out an evaluation which would be a quantitative and qualitative reflection of reality. This important undertaking requires the cooperation of all.

With respect to the learning outcomes, a more immediate and precise answer is expected. Much of the answer can be obtained by considering the objectives.

These are three categories of terminal objectives in the program.

The first category includes objectives which remain stable and are shared by all, regardless of the material and the language used. These objectives are found particularly in the sections which cover General Objectives 4 and 5. They could provide the basis for a common process of evaluation, such as an examination, which would involve the teacher, the school, and the school board, and which could even be provincial in scope.

The second category encompasses a number of objectives which clearly depend on the hardware and, above all, on the programming language selected by the school or the school board. These objectives are found in particular in the section which corresponds to General Objective 3. They too can lead to a traditional examination. However, considering the great diversity of hardware and programming languages available, this evaluation should be the responsibility of the teacher, the school, or the school board.

The third category of objectives is more difficult to evaluate by the traditional methods. These objectives include in particular some of those which develop General Objectives 1, 2, and 6. They do not focus on concepts but rather on acquired attitudes, working methods, or skills to be developed. These terminal objectives cannot be measured solely by the pupil's final product (program, works) or by a traditional exam. It is necessary to observe closely a whole series of intermediate phases which occur between the statement of a problem and its final solution. Consequently, the teacher is the person best qualified to perform this last evaluation. He could use for this purpose an observation form on which the terminal objectives and the strategies applied by the pupil are registered.

SAMPLE OBSERVATION FORM

PUPIL'S NAME : _____

PROJECT : _____

REMARKS : _____

EVALUATION TABLE

1 2 3 4 5 (°°)

OBSERVATION CRITERIA

1. — The pupil has discovered a special aspect of the problem.

2. — The pupil has found a sub-problem.

3. — The pupil uses graphs and diagrams to represent the information contained in the problem.

4. — The pupil selects a systematic approach.

5. — The pupil uses various problem-solving methods.

6. — The pupil accepts the modifications suggested by his classmates or teacher.

(°°) 1. Not at all
2. Rarely
3. Occasionally
4. Often
5. Always

Finally, the social and affective objectives should be evaluated, but not for the purpose of grading the pupil. As is well known, these objectives are particularly difficult to formulate and to evaluate. However, their evaluation is still essential because it provides the teacher with useful information about the nature and the appropriateness of his teaching strategies. It enables him to know more precisely what motivates his pupils and to modify his instructional approach accordingly, thus ensuring more effective learning.

Since the first category of objectives barely covers 10% of the total content of the program, there is no general examination for this program. Evaluation should therefore take place at two levels: first, by means of a traditional test devised by the school board or the school, but involving the teacher; second, by means of a different and very substantial evaluation by the teacher himself. It is suggested that the final grade be averaged thus: 30% for the traditional examination, and 70% for the evaluation done by the teacher.

Obviously evaluation should not be limited to quantitative measurement but should also involve qualitative data. Although the latter dimension of evaluation raises certain difficulties, it is hoped that serious research will one day make available specific means of evaluating these qualitative data. In the meantime, each evaluator should, as much as possible and within reason, produce an evaluation which includes both quantitative and qualitative data.

In conclusion, the importance of evaluation should again be emphasized. By its very nature the program Introduction to Computer Science requires that several items be evaluated: processes, projects, attitudes, behaviors, achievements, values, etc. The following table indicates the need for an evaluative landscape and the neccessity of having both teacher and pupil closely involved in the evaluation process. It is also suggested that all the individuals involved in this program meet to exchange their experiences using the structures and procedures set up by the ministère de l'Éducation.

EVALUATIVE LANDSCAPE

Subjects of the evaluation

EVALUATION	1. PROGRAM	2. TEACHING	3. OUTCOMES
1. Diagnosis		x	x
2. Program	x	x	
3. Comparison	x	x	x
4. Identification of needs	x		
5. Evaluation of attainment of the objectives.	x	x	x

x: indicates that the teacher plays the main role

5
Special Factors

5. Special Factors

5.1 Conditions of Admission

There should not be any restrictive conditions of admission to this program based on psychoeducational criteria. The objectives are sufficiently broad and flexible to accommodate all types of pupils. The intention is to demystify computer science and to contribute to the general education of all pupils.

5.2 Time Allocation

Each course has been assigned four credits. The percentage of time allocated for each general objective is indicated on page 21.

These percentages indicate both the importance of each objective with respect to the others, and the time specifically set aside for a particular objective. In other words, the percentage number is intended to suggest the amount of teaching time to be spent on a given objective, keeping in mind that this amount can vary according to the circumstances. For instance, Objective 4, "To know the fundamental structures and components of a computer system, as well as the history of the computer," has been attributed 4%, which indicates the percentage of the time which should be devoted to its attainment. However, it is obvious that while working on Objective 3, "To program a computer," the pupil indirectly and directly covers Objective 4. That is to say that while he programs the computer he also learns how the computer works and how it has been developed.

The objectives are not isolated units but interrelated and overlapping elements. The program's practical approach to computer science (algorithms, problem-solving, heuristics) makes it possible to cover in a useful and natural manner a number of objectives which are very important (General Objectives 5 and 6 in particular), but whose nature and focus present certain peculiarities. It is important therefore that an adequate hardware be made available to the pupil as soon as possible.

Bibliography

Aiken, Robert M. and Ludwig Brown. "Into the 80's With Microcomputer Based Learning." *Computer*, Vol. 13, No. 7, (July 1980), pp. 11-16.

Aiken, Robert M. and David C. Rine. "Computer Education." *Computer*, Vol. 13, No. 6, (June 1980), pp. 7-9.

Aiken, Robert M. and David C. Rine. "Using Computers in Education." *Computer*, Vol. 13, No. 7, (July 1980), pp. 9-11.

Awad, Elias M. *Business Data Processing*. Englewood Cliffs, N.J.: Prentice-Hall, 1971.

Booth, Taylor L. *Digital Networks and Computer Systems*. New York: John Wiley and Sons, Inc., 1971.

Branstad, Martha A. et al. "Validation, Verification and Testing for the Individual Programmer." *Computer*, Vol. 13, No. 12, (Dec. 1980), pp. 24-30.

Caude, R., A. Moles, et al. *Méthodologie vers une science de l'action*. Paris: Gauthier-Villars Editeur, 1964.

Chedeville, F., F. Faure, and C. Lafond. *Langage symbolique d'enseignement (L.S.E.): manuel d'utilisation*. Paris: Institut National de Recherche et de Documentation, 1975.

Chomsky, N. and J. Piaget. *Théorie du langage. Théories de l'apprentissage*, (Débat organisé et recueilli par Massimo Piattelli-Patmarini, tradult par Yvonne Norzet). Paris: Seuil, 1979.

Control Data Corporation. *Algol Reference Manual*. Sunnyvale, California, 1971.

Control Data Corporation. *Fortran Extended Reference Manual*. Sunnyvale, California, 1971.

Crowley, Thomas H. *Understanding Computers*. New York: McGraw-Hill Book Co., 1967.

De Montgomery, Marie-Pierre. *Le communication homme-machine*. Nice: Centre du XX^{e} Siècle, Université de Nice, 1979.

De Villers, Marie-Eva. *Vocabulaire de l'informatique de gestion, (cahier no. 35)*. Montréal: Office de la Langue Française, 1980.

Dubuc, R. et al. *Dictionnaire de l'informatique*. Québec: Dunod, 1971.

Dufort, J.A., V. Godbout, and C. Paquette. *Introduction à l'heuristique: résolution de problèmes*, (PMM 4002, PERMAMA). Québec: Université du Québec, T.E.L.U.Q., 1976.

Education Board: Joint Issue (SIGCSE, SIGCUE). *Topics: Computer Education for Elementary and Secondary Schools January, 1981*. New York: ACM, 1133 Avenue of the Americas, 1981.

Education Development Center Inc. *Comprehensive Problem Solving in Secondary Schools: A Conference Report. Newton, Mass.* Boston: Houghton Mifflin Co., 1975.

Eisner, Elliot W. *The Educational Imagination: On the Design and Evalution of School Programs*. New York: Macmillan Publishing Co., 1979.

Enos, Judith C. and R.L. Van Tilbury. "Software Design." *Computer*, Vol. 14 No. 2, (Feb. 1981), pp. 61-83.

Feldman, Jérôme. "Les langages de programmation." *Pour la science*, no. 28, (février 1980), pp. 80-91.

Gear, William C. *Computer Organization and Programming*. New York: McGraw-Hill Book Co., 1969.

Ginguay, Michel. *Dictionnaire d'informatique: Français-Anglais*. Paris: Masson et Cie., 1976.

Ginguay, Michel. *Dictionnaire d'informatique: Anglais-Français*. Paris: Masson et Cie., 1979.

Ginguay, M. and A. Lauret. *Lexique d'informatique*. Paris: Masson et Cie., 1973.

Gouvernement du Québec, Ministère de l'Education (Direction des programmes). *Guide général sur l'utilisation de l'ordinateur comme instrument pédagogique au secondaire*. Québec, 1979.

Hewlett-Packard Company. *H.P. 2000 Basic Reference Manual*. Santa Clara, California, 1976.

Hsia, Pei and Frederick E. Petry. "A Systematic Approach to Interactive Programming." *Computer*, Vol. 13, No. 6, (June 1980), pp. 27-34.

The Journal of Children's Mathematical Behavior, Vol. II, No. 1, (Spring 1978).

Korn, Granino A. *Minicomputers for Engineers and Scientists*. New York: McGraw-Hill Book Co., 1973.

Labelle, Marcel, and Alain Taurisson. *Quelques lignes directrices concernant l'évaluation de projets d'élèves*. (Document interne, PERMAMA, Université du Québec, 1979).

Lemos, Ronald S. "Methods, Styles and Attitudes in the Programming Language Classroom." *Computer*, Vol. 13, No. 4, (April 1980), pp. 58-65.

Lewin, Douglas, *Theory and Design of Digital Computers*. London: Thomas Nelson and Sons, Inc., 1972.

Lindsay, Peter H. and Donald A. Norman. *Traitement de l'information et comportement humain: une introduction à la psychologie*. Montréal: Editions Etudes Vivantes, 1980.

Lipson, Joseph, I. "Technology in Science Education: The Next 10 Years." *Computer*, Vol. 13, No. 7 (July 1980), pp. 23-28.

Newell, A. and H.A. Simon. *Human Problem Solving*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1972.

Papert, Seymour. *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books, 1980.

Polya, George. *Comment poser et résoudre un problème*. Traduit par C. Mesnage. Paris: Dunod, 1962.

Polya, George. *La découverte des mathématiques, tome 1: Les modèles, tome 2: Une méthode générale*. Traduit par M. Didier et M. Fraderie. Paris: Dunod, 1967.

Raskin, J. and Whitney, T. "Perspective on Personal Computing." *Computer*, Vol. 14, No. 1, (Feb. 1980), pp. 52-73.

Société Générale Tortue Pro Plus. *Manuel Logo*. Boucherville, Québec, 1977.

Spitzbarth, Laurel M. *Basic Cobol Programming*. Reading, Mass.: Addison-Wesley Publishing Co., 1970.

Wildmann, Clark. *Aplum Reference Manual*. Amherst, Mass.: APL Group, University of Massachusetts Computing Center, 1975.

Glossary

This short glossary of computer science terms is intended to facilitate the reading of this document. These are all terms used in business data processing. For further information the reader should consult a specialized dictionary.

Algol
An international problem-oriented programming language. (ALGOrithmic Language).

Algorithm
The series of organized steps and well-defined rules or processes used to solve a problem.

Alphanumeric
A term used to denote characters which can be processed by the machine. These include the ten decimal digits and the twenty-six letters of the alphabet.

Analog
Refers to any calculating machine or device in which a continuous representation of variable physical data is used as input.
NOTE: Contrasted with digital.

Analysis
(a) Examination of a programming task and the planning of a solution.
NOTE: This process is sometimes referred to as systems analysis.

(b) Detailed definition and planning of a system of procedures for solving a problem on the computer (systems study).

Application
In the context of a data processing system, application refers to a program or programs especially designed to perform or solve specific tasks.
Examples: personnel management, payroll, inventory.

Assembler
A software program written for a given model of computer and designed to translate symbolic language into machine language.

Assembler Language
A symbolic language for a particular model of computer. In the language the instructions and most of the functions are equivalent to machine language instructions or macro-instructions.

Automation
(a) The implementation of processes by automatic means.

(b) Transformation of a technique, a process, an installation, or a piece of equipment in order to make them automatic.

Bit
An abbreviation of binary digit (one of the two digits, 0 and 1) used in binary notation.

Block
A set of binary elements such as characters, words, or digits, grouped for technical purposes and to be handled as a unit.

Branching
A technique used to modify the sequence in which computer instructions are executed.

Byte
A set of adjacent bits, usually eight or sixteen, that is treated as a unit and is used to encode one character.

Central Processing Unit (CPU)
The part of a computer system which contains the logic and arithmetic circuits, the control circuits, and the main memory. The CPU controls the interpretation and execution of instructions.

Cobol
A high-level programming language developed for business data processing applications.
Common Business Oriented Language.

Code
(a) A conventional system of rules and symbols used to convert data from one form of representation into another.

(b) Representation of an item of information by its equivalent in another form.

Coding
The process of writing data and instructions in a particular computer language.

Command
(a) A control signal that starts, stops, or continues some operation.

(b) The portion of the instruction word that specifies the operation to be performed by the computer.

Compiler
A program that translates high-level programming languages to the machine language of a particular computer.

Computer
An automatic information processing machine designed to perform certain operations on data.

Connection
Refers to the linking of two pieces of equipment which perform different functions.
Note: This applies to the type, number, and role of the connecting circuits, as well as to the type and form of the signals exchanged on those circuits.

Console Station
An interactive terminal containing the control knobs and command switches of the computer as well as the control lights and instruments available to the user.
Example: The console may contain, along with a keyboard, a character printer, or a video display screen, thus making it possible for the operator and the machine to interact.

Cybernetics
The discipline which encompasses the theories and techniques pertaining to communications with and control of human beings and machines. The technology behind these devices and machines, equipped with information banks and pseudobrains, enables them to perform their tasks according to certain instructions.

Data
Units of information presented in stylized form and capable of being communicated, interpreted, or processed, either by human beings or by machines.

Data Bank
An organized collection of data files pertaining to a given subject which is usually updated and made available to users according to their particular needs.

Diagram
A graphic or symbolic representation of:
(1) the relationships between the parts of a program,

(2) the relationship between the hardware components comprising a system.

Digital
Refers to the use of numbers or digits for representing data or the processing of such data.
Example: Digital calculator

Note: Contrasted with analog.

Dumping
A technique designed to copy or write the contents of all or part of storage to backup storage.

Editing
To rearrange and validate input data by preparing it for subsequent processing operations and/or output.

File
An organized collection of related items or records which may be processed as a unit.

Flowchart
A diagrammatic representation that uses symbols, arrows and human language to outline the strategy needed for finding a solution to a problem.

Hardware
The physical equipment that comprises a computer system.

High Level Language
A programming language which is not related to any particular type of computer and which is particularly designed for the translation of algorithms into a form understood by a computer. These languages are usually close to those used in particular disciplines (science or business).

Synthetic language.

Information Processing
Systematic execution of a sequence of operations performed on data to derive meaningful results.
Data processing.

Instruction
A set of identifying characters designed to cause a data processing system (computer) to perform certain operations.

In the high level languages, any combination of key words, constants, operators, and variable names which has been assigned a particular meaning in the given language.

Iteration
Repeated execution of a series of instructions.

A programming technique using loops that repeatedly execute a sequence of operations until a particular condition is satisfied.

Language (Programming)
A clearly defined set of characters and related syntax rules for writing computer programs.

Line
1. A horizontal row of characters;
2. A transmission channel between two or more stations;
3. A flowline.

Loading
Inputting data and instructions into the main memory for the program's execution.

Location (Storage)
A basic part of the main memory, which has a fixed length and may be referred to individually by an address.

Loop (or Iteration)
A sequence of instructions that is repeated until a terminal condition prevails.

A sequence or series of instructions ending with a conditional branch instruction which may send the control back to the beginning of the sequence.

Example: A loop may be executed several times in succession until a certain condition is met.

Machine Language
A specific programming language for a given type of computer, in which each instruction can be interpreted and directly executed by the central processing unit.

Memory
A physical location (storage medium) into which data can be entered, in which it can be held, and from which it can be retrieved.

Microprogram
Organized sequence of micro-instructions which together define a specific function that is built into the computer and stored in a special area of the machine. The micro-instructions are translated directly to machine commands and executed.

Operating System
Software made available by the computer manufacturer and including program loaders, language translators, utility programs, user-oriented programs and subprograms, and other operating aids which assist in the overall functioning of the computer.

Operation
A specific action by which any permissible combination of known data is transformed into or new data.

Example: In the addition operation involving the numbers 3 and 5, which gives 8, 3 and 5 are the operands, 8 is the result, and the + sign is the operator which indicates the nature of the operation.

Output
(a) Transfer of information and/or data from the computer's internal storage unit to an external storage medium or an output device.

(b) The visual displaying or printing of these data.

Peripheral
A device attached to a computer, distinct from the central processing unit but controlled by it, that may provide additional capabilities.

Plotter
An output device capable of producing two-dimensional graphic representations of data on display screens, paper tablets, etc.

Process
A sequence of events or actions leading to a given result or having a certain effect.

Program
A series of instructions and signals designed to guide the computer step by step through some problem-solving process.

Program Flowchart
A diagram that represents or describes the sequence and logical relationship between the various steps in a computer program.

Program Library
Organized collection of software stored on a peripheral storage device.

Programmer
A computer specialist who writes programs.

Programming
Writing, testing, and completing programs.

Reader
A device for reading data, prepared outside the computer (off line), into the computer's memory.
Examples: card reader, tape reader, etc.

Recursion
Recursion refers to the ability of a process to call itself.

Screen (Video Display)
A peripheral (usually a cathode ray tube) which displays data.

Secondary Storage (of Information and Data)
Usually an external storage that supplements another storage.
Examples: punched cards, magnetic tapes, magnetic discs.

Simulator
A specific program for one type of computer, designed to translate a program written in the language of another type of computer into the first computer's own language.

Software
The collection of programs, procedures, and rules, and even the related documentation, that may be used on a computer.

Subprogram
A specific and separate set of instructions used by a program to perform a specific task.

Symbolic Language
Any programming language other than a machine language.

Teleprocessing
The processing of information and data using the computer and telecommunication lines.

Terminal
An output and/or input peripheral device or array of devices attached to the central processing unit of a computer by a data transmission line.

Waiting Time
Period of time between the input and the beginning of the actual execution of the program

Word
The minimum number of bits that can be processed in one operation and/or moved to or from memory at one time in a given computer. Most micro-computers have either 8 bit or 16 bit words.

Gouvernement du Québec
Ministère
de l'Éducation